**One to One mapping-**

One object is associated with one object only.

The primary key of one table that becomes the foreign key of another table called as one to one mapping.

Example- one customer has one transection

|  |  |
| --- | --- |
| Customer Table | Transection Table |
| Id(PK) | Transection id(PK) |
| Name | Date |
| Email | Total |
| address |  |
| Transection id(FK) |  |

**Customer.java**

**package** com.test;

**import** javax.persistence.CascadeType;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToOne;

**import** javax.persistence.Table;

@Entity

@Table(name = "customer")

**public** **class** Customer {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

@Column(name = "firstName")

**private** String firstName;

@Column(name = "mobileNumber")

**private** String mobileNumber;

@OneToOne(targetEntity = Transection.**class**, cascade = CascadeType.***ALL***)

**private** Transection transection;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getFirstName() {

**return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getMobileNumber() {

**return** mobileNumber;

}

**public** **void** setMobileNumber(String mobileNumber) {

**this**.mobileNumber = mobileNumber;

}

**public** Transection getTransection() {

**return** transection;

}

**public** **void** setTransection(Transection transection) {

**this**.transection = transection;

}

}

**Transection.java**

**package** com.test;

**import** java.util.Date;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.Table;

@Entity

@Table(name = "Transection")

**public** **class** Transection {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

@Column(name = "date")

**private** Date date;

@Column(name = "total")

**private** **int** total;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** **int** getTotal() {

**return** total;

}

**public** **void** setTotal(**int** total) {

**this**.total = total;

}

}

**MainTest.Java**

**package** com.test;

**import** java.util.Date;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Configuration configuration = **new** Configuration();

configuration.configure("hibernate.cfg.xml");

SessionFactory sessionFactory = configuration.~~buildSessionFactory~~();

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

Customer customer = **new** Customer();

customer.setFirstName("sohan");

customer.setMobileNumber("7575858595");

Transection transection = **new** Transection();

transection.setDate(**new** Date());

transection.setTotal(5000);

customer.setTransection(transection);

session.save(customer);

transaction.commit();

session.close();

sessionFactory.close();

}

}

**Hibernate.cfg.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration SYSTEM

"classpath://org/hibernate/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/sys</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"hbm2ddl.auto"*>create</property>

<property name=*"show\_sql"*>true</property>

<mapping class=*"com.test.Customer"*></mapping>

<mapping class=*"com.test.Transection"*></mapping>

</session-factory>

</hibernate-configuration>

**Pom.xml**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>map</groupId>

<artifactId>OneToOneMapping</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.5.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.21</version>

</dependency>

</dependencies>

</project>

**Console output-**

Hibernate: alter table customer drop foreign key FK\_hafk3u74420er4hbe3vd1uyqj

Hibernate: drop table if exists Transection

Hibernate: drop table if exists customer

Hibernate: create table Transection (id integer not null auto\_increment, date datetime, total integer, primary key (id))

Hibernate: create table customer (id integer not null auto\_increment, firstName varchar(255), mobileNumber varchar(255), transection\_id integer, primary key (id))

Hibernate: alter table customer add constraint FK\_hafk3u74420er4hbe3vd1uyqj foreign key (transection\_id) references Transection (id)

Aug 09, 2021 11:23:54 AM org.hibernate.tool.hbm2ddl.SchemaExport execute

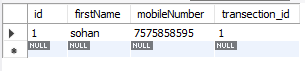
INFO: HHH000230: Schema export complete

Hibernate: insert into Transection (date, total) values (?, ?)

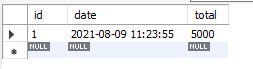
Hibernate: insert into customer (firstName, mobileNumber, transection\_id) values (?, ?, ?)

**Database Table**

Customer table



Transection table



**One to Many mapping-**

One to many mapping means that one row in a table can be mapped to multiple rows in another table.

Example- one user has multiple policies.

|  |  |
| --- | --- |
| User Table | Policy Table |
| Id(PK) | Id(PK) |
| Name | Policyid |
| Email | Userid(FK) |
|  | Status |
|  | Type |

**User.java**

**package** com.example;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

@Entity

@Table(name = "user")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

@Column(name = "name")

**private** String name;

@Column(name = "email")

**private** String email;

@OneToMany(mappedBy = "user")

**private** Set<Policy> policy;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** Set<Policy> getPolicy() {

**return** policy;

}

**public** **void** setPolicy(Set<Policy> policy) {

**this**.policy = policy;

}

}

**Policy.java**

**package** com.example;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.Table;

**import** org.hibernate.annotations.ManyToAny;

@Entity

@Table(name = "policy")

**public** **class** Policy {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** id;

@Column(name = "policyId")

**private** String policyId;

@Column(name = "policyType")

**private** String policyType;

@Column(name = "status")

**private** String status;

@ManyToOne

@JoinColumn(name = "user\_id")

**private** User user;

**public** Policy(String policyId, String policyType, String status, User user) {

**super**();

**this**.policyId = policyId;

**this**.policyType = policyType;

**this**.status = status;

**this**.user = user;

}

}

**Test.java**

**package** com.example;

**import** java.util.HashSet;

**import** java.util.Set;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.hibernate.Transaction;

**import** org.hibernate.cfg.Configuration;

**public** **class** Test {

**public** **static** Configuration getConfiguration() {

Configuration configuration = **new** Configuration();

configuration.configure("hibernate.cfg.xml");

**return** configuration;

}

**public** **static** **void** main(String[] args) {

Configuration configuration = *getConfiguration*();

SessionFactory sessionFactory = configuration.~~buildSessionFactory~~();

Session session = sessionFactory.openSession();

Transaction transaction = session.beginTransaction();

User user = **new** User();

// pass the value through constructor

Policy policy1 = **new** Policy("J5", "Car Insurance", "Active", user);

Policy policy2 = **new** Policy("J6", "Bike Insurance", "Active", user);

Set<Policy> set = **new** HashSet<Policy>();

set.add(policy1);

set.add(policy2);

user.setPolicy(set);

user.setName("ajay");

user.setEmail("ajay@gmail.com");

session.save(user);

session.save(policy1);

session.save(policy2);

transaction.commit();

session.close();

sessionFactory.close();

}

}

**hibernate.cfg.xml**

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE hibernate-configuration SYSTEM

"classpath://org/hibernate/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name=*"hibernate.connection.driver\_class"*>com.mysql.jdbc.Driver</property>

<property name=*"hibernate.connection.url"*>jdbc:mysql://localhost:3306/sys</property>

<property name=*"hibernate.connection.username"*>root</property>

<property name=*"hibernate.connection.password"*>root</property>

<property name=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</property>

<property name=*"hbm2ddl.auto"*>create</property>

<property name=*"show\_sql"*>true</property>

<mapping class=*"com.test.User"*></mapping>

<mapping class=*"com.test.Policy"*></mapping>

</session-factory>

</hibernate-configuration>

**Pom.xml**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>map</groupId>

<artifactId>OneToManyMapping</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>4.3.5.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.21</version>

</dependency>

</dependencies>

</project>

**Console output-**

Hibernate: alter table policy drop foreign key FK\_nyh2d928sehpgfhlwjl7ohqt8

Hibernate: drop table if exists policy

Hibernate: drop table if exists user

Hibernate: create table policy (id integer not null auto\_increment, policyId varchar(255), policyType varchar(255), status varchar(255), user\_id integer, primary key (id))

Hibernate: create table user (id integer not null auto\_increment, email varchar(255), name varchar(255), primary key (id))

Hibernate: alter table policy add constraint FK\_nyh2d928sehpgfhlwjl7ohqt8 foreign key (user\_id) references user (id)

Aug 09, 2021 11:32:44 AM org.hibernate.tool.hbm2ddl.SchemaExport execute

INFO: HHH000230: Schema export complete

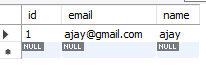
Hibernate: insert into user (email, name) values (?, ?)

Hibernate: insert into policy (policyId, policyType, status, user\_id) values (?, ?, ?, ?)

Hibernate: insert into policy (policyId, policyType, status, user\_id) values (?, ?, ?, ?)

**Database Table-**

User table



Policy table

